HEWLETT PACKARD

HP-71

Users' Library Solutions

Utilities

CONTENTS

The subprograms provided in this section are intended to function as enhancements to the HP-71 BASIC language. The majority of the routines are complementary to HP-71 keywords. For example, the keyword PWIDTH sets the maximum number of characters per line of PRINT'ed output. The subprogram PWIDTH returns the current PWIDTH setting. With such a subprogram/keyword pair, the user can write a program that queries the HP-71 for the current PWIDTH, saves the value, alters the PWIDTH for a program-specific reason, and returns the PWIDTH to its previous state when finished.

Using the Subprograms

Each subprogram is intended to be used in a fashion similar to that of a standard BASIC keyword. Once a subprogram is loaded into HP-71 memory, it can be called from the keyboard or from a user's program.

EXAMPLE:

From the keyboard,

```
DIM A$
A$="aBcDeFgH"
CALL LWRCS(A$)
A$              displays "abcdefgh".
```

In a program,

```
10 DIM A$[84]
20 LINPUT 'Your name ? ';A$
30 CALL LWRCS(A$)
40 DISP A$
50 END
```

The subprograms do little or no error checking. Those programs that take parameters assume that their values are correct. Neither is there any error trapping: those errors that occur will suspend the program.

Loading the Subprograms

Loading the subprograms into the HP-71 is a straight-forward operation which involves copying the code from the provided listing into your computer. It must be stressed that the code as it appears in HP-71 memory must, in most instances, exactly match the provided listing. This is especially true with the POKE statement. The parameters for POKE must be identical to those given in the listing or disastrous results (including a memory reset) may occur.

The listing contains 23 different and independent subprograms. It is obvious that few users will have need of every subprogram provided. They have therefore been written so that the user may extract any subprogram or group of subprograms desired. All that is involved is to identify the desired subprogram(s) and copy the code from SUB to END

SUB. The line numbering is not important, as long as the lines are ordered in the computer as they are in the listing.

"GSU"

Subprogram Descriptions

CONTRAST(C)	Returns the current contrast setting (0-15) to simple numeric variable C.
CMDSTK(C)	Returns the current maximum number of levels in the command stack (1-16) to simple numeric variable C.
DELAY(D,C)	Returns the row-wise delay parameter to D and the column-wise (character) delay to C. The values returned are in the range 0 to 7 + 30/32, in increments of 1/32, and INF.
DISPLAY(D$)	Returns the numeric display format to D$ in the form "STD", "FIX n", "SCI n", or "ENG n". The dimensioned length of D$ must be greater than or equal to the length of the string to be returned or a fatal error (program suspension) will result. The maximum length of the returned string is 6 characters. n is an integer from 0 to 11.
ENDLINE(E$)	Returns the current PRINT end-of-line sequence (1 to 3 characters). Note that the string returned may not (and probably will not) be DISP'layable or PRINT'able. The dimensioned length of E$ must be greater than or equal to the length of the string to be returned or a fatal error will result.
LTRIM(L$)	Removes leading (left-hand) spaces from a string variable.
LWRCS(L$)	Transforms all uppercase characters in a string variable to lowercase.
OPTROUND(O$)	Returns the numeric rounding mode in the form "NEAR", "NEG", "POS", or "ZERO". The character dimension of O$ must be greater than or equal to the length of the string to be returned or a fatal error (program suspension) will result. The maximum length of the returned string is 4 characters.
PWIDTH(P)	Returns the current maximum-length of PRINT'ed lines: the parameter of the PWIDTH command.
REPEAT(R$,N,C$)	Fills C$ with N copies of the R$. C$ must be dimensioned to a length that is greater than or equal to N times the length of R$. For example, CALL REPEAT('HI !',9,C$) requires C$ to be dimensioned to at least 9*4, or 36 (DIM C$[36]).
REVERSE(R$)	Reverses the order of the characters in R$.

RTRIM(R$)	Removes trailing (right-hand) spaces from a string variable.
SETDISP(S$)	Sets or resets the display mode given a string of the form generated by the DISPLAY(D$) subprogram.
SETROUND(S$)	Sets or resets the rounding mode given a string of the form generated by the OPTROUND(O$) subprogram.
SETCMDST(X)	Sets the maximum number of levels in the command stack to the value of X (from 1 to 16). Values of X that are less than 1 will be taken as 1. Values of X that are greater than 16 will be taken as 16. The command stack will be empty after execution of this subprogram.
SETSYSFL(S$)	Sets or resets the states of all alterable system flags given a string of the form returned by the SYSFLAG(S$) subprogram. S$ may be of any length though only the first 8 characters will be used by the subprogram. Strings of less than 8 characters will affect only the higher numbered (closer to 0) flags.
SETUSRFL(S$)	Sets or resets the states of all user flags given a string of the form returned by the USERFLAG(S$) subprogram. S$ may be of any length though only the first 16 characters will be used by the subprogram. Strings of less than 16 characters will affect only the lower numbered (closer to 0) flags.
STAT(S$)	Returns the current statistical array name. S$ must be dimensioned to a length that is greater than or equal to the length of the returned string or a fatal error will occur. The maximum length of the returned string is 2.
SYSFLAG(S$)	Returns the PEEK$ of the system-flag memory. S$ must have a character dimension of at least 8.
USERFLAG(U$)	Returns the PEEK$ of the user-flag memory. U$ must have a character dimension of at least 16.
WIDTH(W)	Returns the current maximum display width: the parameter of the WIDTH statement.
WINDOW(W,L)	Returns the current window start (W) and end (L): parameters of the WINDOW statement.
ZAPSPCS(Z$)	Removes all spaces from a string variable.

Listing

"GSU"

```
10 SUB CONTRAST(C)
20 C=HTD(PEEK$("2E3FE",1))
30 END SUB
40 !
50 SUB CMDSTK(C)
60 C=HTD(PEEK$("2F976",1))+1
70 END SUB
80 !
90 SUB DELAY(D,C)
100 A$=PEEK$("2F946",4)
110 DEF FNR(B$)
120 IF B$="FF" THEN FNR=INF ELSE FNR=HTD(B$)/32
130 END DEF
140 D=FNR(A$[4]&A$[3,3])
150 C=FNR(A$[2,2]&A$[1,1])
160 END SUB
170 !
180 SUB DISPLAY(D$)
190 B$=PEEK$("2F6DC",2)
200 D$="STDFIXSCIENG"[MOD(HTD(B$[1,1]),4)*3+1][1,3]
210 STD
220 IF D$#"STD" THEN D$=D$&" "&STR$(HTD(B$[2]))
230 POKE "2F6DC",B$
240 END SUB
250 !
260 SUB ENDLINE(E$)
270 E$=""
280 L=HTD(PEEK$("2F95A",1))
290 A$=PEEK$("2F95B",L)
300 FOR I=2 TO L STEP 2
310 E$=E$&CHR$(HTD(A$[I,I]&A$[I-1,I-1]))
320 NEXT I
330 END SUB
340 !
350 SUB TRIMLEFT(L$)
360 FOR I=1 TO LEN(L$)
370 IF L$[I,I]#" " THEN 'DONE'
380 NEXT I
390 'DONE': L$=L$[I]
400 END SUB
410 !
420 SUB LWRCS(L$)
430 FOR I=1 TO LEN(L$)
440 IF NUM(L$[I])>64 AND NUM(L$[I])<91 THEN L$[I,I]=CHR$(NUM(L$[I])+32)
450 NEXT I
460 END SUB
470 !
480 SUB OPTROUND(D$)
490 D$="NEARPOS ZERONEG "[HTD(PEEK$("2F6DB",1)) DIV 4*4+1][1,4]
500 IF D$[4]=" " THEN D$=D$[1,3]
510 END SUB
520 !
530 SUB PWIDTH(P)
540 A$=PEEK$("2F958",2)
```

```
550 P=HTD(A$[2]&A$[1,1])
560 IF NOT P THEN P=INF
570 END SUB
580 !
590 SUB REPEAT(R$,N,C$)
600 C$=""
610 FOR I=1 TO N
620 C$=R$&C$
630 NEXT I
640 END SUB
650 !
660 SUB REVERSE(A$)
670 L=LEN(A$) @ DIM D$[L]
680 FOR I=L TO 1 STEP -1
690 D$=D$&A$[I,I]
700 NEXT I
710 A$=D$
720 END SUB
730 !
740 SUB RTRIM(A$)
750 FOR I=LEN(A$) TO 1 STEP -1
760 IF A$[I,I]#" " THEN 'DONE'
770 NEXT I
780 'DONE': A$=A$[1,I]
790 END SUB
800 !
810 SUB SETDISP(A$)
820 GOTO A$[2,2]
830 'I': FIX VAL(A$[4]) @ GOTO 'DONE'
840 'C': SCI VAL(A$[4]) @ GOTO 'DONE'
850 'N': ENG VAL(A$[4]) @ GOTO 'DONE'
860 'T': STD
870 'DONE': END SUB
880 !
890 SUB SETROUND(C$)
900 A$=PEEK$("2F6DB",1)
910 A$=DTH$(MOD(HTD(A$),4)+(POS("NEARPOS ZERONEG ",C$)-1))
920 POKE "2F6DB",A$[5,5]
930 END SUB
940 !
950 SUB SETCMDST(X)
960 DEF FNR$(A$)=A$[5]&A$[4,4]&A$[3,3]&A$[2,2]&A$[1,1]
970 IF X<1 THEN X=1
980 IF X>16 THEN X=16
990 DIM S$[X*6]
1000 A=HTD(FNR$(PEEK$("2F576",5)))
1010 FOR I=1 TO X
1020 S$=S$&"000300"
1030 NEXT I
1040 E$=FNR$(DTH$(A+X*6))
1050 POKE "2F580",E$&E$&E$
1060 POKE DTH$(A),S$
1070 POKE "2F976",DTH$(X-1)[5,5]
1080 END SUB
1090 !
1100 SUB SETSYSFL(S$)
1110 POKE "2F6D9",S$[1,8]
```

```
1120 END SUB
1130 !
1140 SUB SETUSRFL(U$)
1150 POKE "2F6E9",U$[1,16]
1160 END SUB
1170 !
1180 SUB STAT(S$)
1190 A$=PEEK$("2F7AD",3)
1200 S$=(CHR$(HTD(A$[2,2]&A$[1,1]))&STR$(HTD(A$[3])-1))[1,2-NOT HTD(A$
[3])]
1210 END SUB
1220 !
1230 SUB SYSFLAG(S$)
1240 S$=PEEK$("2F6D9",8)
1250 END SUB
1260 !
1270 SUB USERFLAG(U$)
1280 U$=PEEK$("2F6E9",16)
1290 END SUB
1300 !
1310 SUB WIDTH(W)
1320 A$=PEEK$("2F94F",2)
1330 W=HTD(A$[2]&A$[1,1])
1340 IF NOT W THEN W=INF
1350 END SUB
1360 !
1370 SUB WINDOW(W,L)
1380 A$=PEEK$("2F471",4)
1390 W=HTD(A$[2,2]&A$[1,1])+1
1400 L=W+HTD(A$[4]&A$[3,3])
1410 END SUB
1420 !
1430 SUB ZAPSPCS(Z$)
1440 FOR I=1 TO LEN(Z$)
1450 'MORE': IF Z$[I,I]=" " THEN Z$[I,I+1]="" @ GOTO 'MORE'
1460 NEXT I
1470 END SUB
```

The purpose of these routines is to allow the user to perform numeric calculations based on time and date.

Using the Subprograms

The subroutines are designed to work on numeric data - numbers that represent time or date. Time numbers are of the form HH.MMSSDDDD, where the integer portion of the number is the number of hours, the first two digits (MM) of the fractional portion are the minutes, the second two digits (SS) are the number of seconds, and all trailing digits (DDDD) are fractions of seconds. Date numbers are of the form YYYYMMDD, where YYYY is the four-digit year, DD is the two-digit month, and DD is the two digit day. Subprograms are also provided that convert between these numbers and formatted time and date strings.

Each subprogram is intended to be used in a fashion similar to that of a standard BASIC keyword. Once a subprogram is loaded into HP-71 memory, it can be called from the keyboard or from a user's program.

EXAMPLE:

From the keyboard,

```
DIM T,T$
STD
CALL HMSPLUS(12.235612,5.45281,T)
T                          displays 18.092422
CALL TIMEFMT(T,T$,12)
T$                         displays 06:09:24.22 PM
```

In a program,

```
10 DIM S,W,T$
20 STD
30 INPUT "Start, Worked ? ";S,W
40 CALL HMSPLUS(S,W,T)
50 CALL TIMEFMT(T,T$,12)
60 DISP T$;" when done."
70 END
```

The subprograms do little or no error checking. Those programs that take parameters assume that their values are correct. Neither is there any error trapping: those errors that occur will suspend the program.

Loading the Subprograms.

Loading the subprograms into the HP-71 is a straight-forward operation which involves copying the code from the provided listing into your computer. It must be stressed that the code as it appears in HP-71 memory must, in most instances, exactly match the provided listing. This is especially true with the POKE statement. The parameters for

POKE must be identical to those given in the listing or disastrous results (memory reset) may occur.

The listing contains 13 different subprograms. It is obvious that few users will have need of every subprogram provided. Most have therefore been written so that the user may extract the subprogram or group of subprograms desired. The exceptions are SUB DOW, which requires SUB JULIAN; SUB DDAYS also requires SUB JULIAN; and SUB DATEPLUS, which requires both SUB JULIAN and SUB DATE. For all other subprograms, all that is involved is for the user to identify the desired subprogram(s) and copy the code from SUB to END SUB, inclusive. The line numbering is not important, as long as the lines are ordered in the computer as they are in the listing.

"TMDTUT"

Subprogram Descriptions

HMS(H1,H2) Converts a number whose integer portion represents whole hours (or degrees) and whose fractional portion is fractional hours (or degrees) to a number of the form H.MMSSDDDD where H is whole hours, MM is number of minutes, SS is number of seconds and DDDD is fractional seconds. Use of INTEGER variables for H1 and/or H2 may cause loss of significant information through rounding.

HR(H1,H2) Converts a number of the form H.MMSSDDDD where H is whole hours, MM is number of minutes, SS is number of seconds and DDDD is fractional seconds to a number whose integer portion represents whole hours (or degrees) and whose fractional portion is fractional hours (or degrees). Use of INTEGER variables for H1 and/or H2 may cause loss of significant information through rounding.

HMSPLUS(H1,H2,H3) Adds two HMS formatted numbers to get a third (H1 and H2 to get H3). Use of INTEGER variables for H1, H2 and/or H3 may cause loss of significant information through rounding.

TIMEFMT(T,T$,F) Converts an HMS formatted number to a time string of the form "HH:MM:SS.DDDD". The format (F, either 12 or 24) determines whether HH is represented in twelve or twenty four hour format. If the format is 12, " AM", " PM", " MDNT" or " NOON" will be appended to the end of the string, and if the integer portion of the number is greater than 12, the integer portion of the output will have the appropriate multiple of 12 subtracted from it so that it is 12 or less. A similar statement can be made using 24 in 24-hour format. The precision of the fractional portion of seconds corresponds to the value selected by the user via FIX, SCI or ENG, though the displayed value will be in FIX'ed format.

The length of T$ must be greater than or equal to the length of the returned time string or a fatal error will occur. The length of the string returned varies with the display format and the time number. A length of 34 characters is possible under extreme circumstances, while a string length of less than 25 characters is more common.

JULIAN(D,J) Converts a date formatted number (YYYYMMDD) into its Julian day number. JDN's are valid only from 1/1/4713 BC to any future date where the current dating system can be assumed to be in use. Dates before 10/4/1582 are considered from the Julian calendar while those after are considered from the Gregorian calendar. J must be a REAL variable.

DATE(J,D) Converts a Julian day number into a date formatted number (YYYYMMDD). JDN's are valid only from 1/1/4713 BC to any future date where the current dating system can be assumed to be in use. Dates before 2445612 are considered from the Julian calendar while those after are considered from the Gregorian calendar. D must be a REAL variable.

DOW(D,D$) Returns the day of week for a given calendar date, where the date is given in YYYYMMDD format. The character dimension of D$ must be greater than or equal to the length of the returned string or a fatal error will result. The maximum length of the returned string is 9 characters. This subprogram requires SUB JULIAN.

DOY(D1,D2) Returns the day of year for the given date. Both values are in YYYYMMDD format.

DDAYS(D1,D2,N) Returns the number of days between two dates. Both dates are expressed in YYYYMMDD format. This subprogram requires SUB JULIAN.

DATEPLUS(D1,N,D2) Returns the date number N days different from D1. Both dates are expressed in YYYYMMDD format. The number N is a positive or negative integer. This subprogram requires SUB JULIAN and SUB DATE.

TIMENUM(T$,T) Converts a timestring of the format "HH:MM:SS.DDDD" to an HMS formatted time number. The suffixes " AM" and " PM" are optional. If " PM" is appended to the time string and the numeric time depicted is less than 12, 12 hours will be added to the output (e.g., "12:59:59 PM"). Midnight is considered to be 24:00:00 and not 00:00:00. Therefore, T$="00:00:01 MDNT" will not progduce valid results. Use of an INTEGER variable for T may cause loss of significant information through rounding.

DATEFMT(D,D$,F$) Converts a datenumber into a date string of the form "MM/DD/YYYY" or "DD.MM.YYYY" based on the

format specifier (F$, either "MDY" or "DMY" respectively). If the input date string is negative, " BC" is appended to the date string.

The length of D$ must be greater than or equal to the length of the returned date string or a fatal error will occur. The length of the string returned varies with the date number. The maximum (reasonable) length of the returned string is 13 characters.

DATENUM(D$,D) Converts datestrings of the format "MM/DD/YYYY" or "DD.MM.YYYY" to a datenumber. The suffixes " BC" and " AD" are optional. If the suffix " BC" is used, the number returned will be negative.

TMDT

Listing

```
10 SUB HMS(H1,H2)
20 H2=ABS(FP(H1))*60
30 H2=SGN(H1)*(ABS(IP(H1))+IP(H2)/100+FP(H2)*.006)
40 END SUB
50 !
60 SUB HR(H1,H2)
70 M=ABS(FP(H1)*100)
80 H2=SGN(H1)*(ABS(IP(H1))+IP(M)/60+FP(M)/36)
90 END SUB
100 !
110 SUB HMSPLUS(H1,H2,H3)
120 H3=(IP(H1)+IP(H2))*3600
130 H3=H3+(IP(FP(H1)*100)+IP(FP(H2)*100))*60
140 H3=H3+(FP(H1*100)+FP(H2*100))*100
150 H=H3 DIV 3600
160 M=RMD(H3,3600) DIV 60
170 S=RMD(H3,60)
180 H3=H+M/100+S/10000
190 END SUB
200 !
210 SUB TIMEFMT(T,T$,F)
220 DEF FNZ$(A)
230 IF A<10 THEN FNZ$="0"&STR$(A) ELSE FNZ$=STR$(A)
240 END DEF
250 S0=SGN(T) @ T=ABS(T)
260 F=F=12
270 D$=PEEK$("2F6DC",2)
280 M=IP(FP(T)*100)
290 S=FP(T*100)*100
300 STD
310 T0=RMD(IP(T),24-12*F)
320 IF T0=0 THEN T0=T0+24-12*F
330 T$=FNZ$(T0)&":"&FNZ$(M)&":"
340 IF MOD(HTD(D$[1,1]),4) THEN FIX HTD(D$[2])
350 T$=T$&FNZ$(S)
360 POKE "2F6DC",D$
370 IF S0<0 THEN T$="-"&T$
380 IF NOT F THEN "E"
390 IF NOT RMD(T,24) THEN T$=T$&" MDNT" @ GOTO "E"
400 IF NOT RMD(T,12) THEN T$=T$&" NOON" @ GOTO "E"
410 IF RMD(IP(T) DIV 12,2) THEN T$=T$&" PM" ELSE T$=T$&" AM"
420 'E': END SUB
430 !
440 SUB JULIAN(D,J)
450 B=D<0
460 M=MOD(ABS(D),10000) DIV 100
470 Y=D DIV 10000+B
480 IF M<3 THEN M=M+12 @ Y=Y-1
490 D0=RMD(ABS(D),100)
500 J=IP(365.25*Y-B*.75)+IP(30.6001*(M+1))+D0+1720995
510 IF D>15821004 THEN J=J-Y DIV 100+Y DIV 400+2
520 END SUB
530 !
540 SUB DATE(J,D)
550 J0=J<2199161
560 C=(J+68569+J0*38) DIV (36524.25+J0*.75)-49
```

```
570 A=IP(MOD(J+68569+J0*38,36524.25+J0*.75))
580 D=(A+1) DIV 365.25025
590 B=A-IP(365.25*D)
600 M=(B+31) DIV 30.59
610 D0=B+31-IP(30.59*M)
620 Y=100*C+D+M DIV 11
630 IF J<1721421 THEN Y=ABS(Y-1)
640 IF M<11 THEN M=M+2 ELSE M=M-10
650 D=Y*10000+M*100+D0
660 IF J<1721421 THEN D=-D
670 END SUB
680 !
690 SUB DOW(D,D$)
700 CALL JULIAN(D,J)
710 D$="Mon....Tues...Wednes.Thurs..Fri....Satur..Sun."[RMD(J,7)*7+1][
1,7]
720 D$[POS(D$,".")]="day"
730 END SUB
740 !
750 SUB DOY(D1,D2)
760 M=MOD(ABS(D1),10000) DIV 100
770 Y=ABS(D1) DIV 10000
780 D=RMD(ABS(D1),100)
790 IF D1>0 THEN L=RMD(Y,4)=0 ELSE L=RMD(Y,4)=1
800 IF M<3 THEN D2=(M-1)*(63-L) DIV 2 ELSE D2=IP((M+1)*30.6)-63+L
810 D2=D2+D
820 END SUB
830 !
840 SUB DDAYS(D1,D2,N)
850 CALL JULIAN(D1,J1)
860 CALL JULIAN(D2,J2)
870 N=J2-J1
880 END SUB
890 !
900 SUB DATEPLUS(D1,N,D2)
910 CALL JULIAN(D1,J)
920 CALL DATE(J+N,D2)
930 END SUB
940 !
950 SUB TIMENUM(T$,T)
960 C=POS(T$,":")
970 S=SGN(VAL(T$)) @ IF NOT S THEN S=1
980 T=S*ABS(VAL(T$)+VAL(T$[C+1])/100+VAL(T$[POS(T$,":",C+1)+1])/10000)
990 IF (POS(UPRC$(T$),"PM") OR POS(UPRC$(T$),"MDNT")) AND ABS(T)<12 TH
EN T=SGN(T)*(ABS(T)+12)
1000 END SUB
1010 !
1020 SUB DATEFMT(D,D$,F$)
1030 DEF FNZ$(A)
1040 IF A<10 THEN FNZ$="0"&STR$(A) ELSE FNZ$=STR$(A)
1050 END DEF
1060 S0=SGN(D) @ D=ABS(D)
1070 D0$=PEEK$("2F6DC",2)
1080 M=RMD(D,10000) DIV 100 @ D0=RMD(D,100) @ Y=D DIV 10000
1090 IF UPRC$(F$)="DMY" THEN T=M @ M=D0 @ D0=T @ C$="." ELSE C$="/"
1100 STD
1110 D$=FNZ$(M)&C$&FNZ$(D0)&C$
```

```
1120 IF Y<1000 THEN D$=D$&"0"
1130 IF Y<100 THEN D$=D$&"0"
1140 D$=D$&FNZ$(Y)
1150 POKE "2F6DC",D0$
1160 IF S0<0 THEN D$=D$&" BC"
1170 END SUB
1180 !
1190 SUB DATENUM(D$,D)
1200 IF POS(D$,"/") THEN C$="/" ELSE C$="."
1210 C=POS(D$,C$)
1220 D=ABS(VAL(D$[1,C-1]))
1230 M=ABS(VAL(D$[C+1,POS(D$,C$,C+1)-1]))
1240 Y=ABS(VAL(D$[POS(D$,C$,C+1)+1]))
1250 IF POS(D$,"/") THEN T=D @ D=M @ M=T
1260 D=D+M*100+Y*10000
1270 IF POS(UPRC$(D$),"BC") THEN D=-D
1280 END SUB
```

The purpose of these subprograms is to provide a base of generic conversions that can be used both from the keyboard and from within a program. The bulk of the routines are concerned with the relatively common problem of conversions between decimal feet and feet-inches-fractions of inches. Subprograms have been provided to allow both conversion to and from decimal feet and feet-inches and to allow addition, subtraction, multiplication and division of feet-inch formatted numbers.

A separate, generic, conversion subprogram has also been provided to allow conversions between 81 common units. The routine is relatively sophistocated in that it not only allows unit-to-unit conversions (viz., miles to kilometers) but will also evaluate expressions in units (viz., mi/hr/min to m/s^2). The subprogram is also relatively easy to modify, so that other commonly used conversion units may be added or superfluous units deleted from its list.

Using the Subprograms

Each subprogram is intended to be used in a fashion similar to that of a standard BASIC keyword. Once a subprogram is loaded into HP-71 memory, it can be called from the keyboard or from a user's program.

EXAMPLE:

From the keyboard,

```
DIM F$
SCI 5
CALL FEETADD("12 2 56/64","5 4 28/32",64,F$)
F$                          displays 17 7 48/64
```

In a program,

```
10 DIM A$,B$,O$
20 STD
30 INPUT "Lenght A ? ";A$
40 INPUT "Lenght B ? ";B$
50 CALL FEETADD(A$,B$,32,O$)
60 DISP O$;" total ft."
70 END
```

The subprograms do little or no error checking. Those programs that take parameters assume that their values are correct. Neither is there any error trapping: those errors that occur will suspend the program.

Loading the Subprograms.

Loading the subprograms into the HP-71 is a straight-forward operation which involves copying the code from the provided listing into your computer. It must be stressed that the code as it appears in HP-71

memory must, in most instances, exactly match the provided listing. This is especially true with the POKE statement. The parameters for POKE must be identical to those given in the listing or disastrous results may occur.

The listing contains 8 different subprograms. SUB FEETINCH and SUB DECFEET are independent of all other subprograms in the listing. These two subprograms may be extracted from the listing and used in independent programs. All that is involved is for the user to copy the code from SUB to END SUB, inclusive. The line numbering is not important, as long as the lines are ordered in the computer as they are in the listing. All other feet conversion subprograms are dependent on these two. Therefore, if any of the others is to be used, both SUB FEETINCH and SUB DECFEET must also be present in HP-71 memory.

SUB CONVERT is semi-independent. That is to say, it requires SUB MAKE, but only sometimes. SUB CONVERT requires the existence in memory of data file CONST and text file NAMES. If the files exist, SUB MAKE is not required. If they do not exist, SUB MAKE is called to generate them.

"CALL" (Conversion Utilities

Subprogram Descriptions

FEETINCH(F,F1,F$)

Convert decimal feet (F) to feet, inches and fractions (F$) using the specified denominator (F1). The format of F$ is FF II ff/F1, were FF is the number of feet, II is the number of inches, ff is the number of fractions, and F1 is the denominator of the fraction. For example, CALL FEETINCH(12.345,16,F$) sets F$ to "12 4 2/16", or 12 feet 4 and 2/16 inches. Note that expressing this number in sixteenths of inches causes a loss of accuracy due to rounding to the nearest whole sixteenth. The exact result is 12 feet 4 and 7/50 inches. The character dimension of F$ must be large enough to contain the string returned by the program or a fatal error (program suspended) will result. The length of the returned string varies with the value of F.

DECFEET(F$,F)

Convert a compound number in feet and inches to decimal feet. F$ must be in the format returned by SUB FEETINCH and F must be REAL. Example: CALL DECFEET('12 4 2/16',F) sets F to 12.34375.

FEETADD(F1$,F2$,F0,F3$)

Add two compound numbers in feet and inches (F1$ and F2$) to get a third (F3$). F0 is the denominator in which the fractional portion of F3$ will be expressed. The character dimension of F3$ must be large

enough to contain the string returned by the program or a fatal error (program suspended) will result. The length of the returned string varies with the values of F1$ and F2$.

FEETSUB(F1$,F2$,F0,F3$) Subtract two compound numbers in feet and inches (F1$ and F2$) to get a third (F3$). F0 is the denominator in which the fractional portion of F3$ will be expressed. The character dimension of F3$ must be large enough to contain the string returned by the program or a fatal error (program suspended) will result. The length of the returned string varies with the values of F1$ and F2$.

FEETMULT(F1$,F2$,F0,F3$) Multiply two compound numbers in feet and inches (F1$ and F2$) to get a third (F3$). F0 is the denominator in which the fractional portion of F3$ will be expressed. The character dimension of F3$ must be large enough to contain the string returned by the program or a fatal error (program suspended) will result. The length of the returned string varies with the values of F1$ and F2$.

FEETDIV(F1$,F2$,F0,F3$) Divide two compound numbers in feet and inches (F1$ and F2$) to get a third (F3$). F0 is the denominator in which the fractional portion of F3$ will be expressed. The character dimension of F3$ must be large enough to contain the string returned by the program or a fatal error (program suspended) will result. The length of the returned string varies with the values of F1$ and F2$.

CONVERT(N1,N2,U$) Convert N1 to N2 based on U$. U$ may be in one of three different formats: units; units1-units2; -units. In the first case, N1 is assumed to have the units expressed in U$, and is converted to a number with SI units which is returned to N2. In the second case, N1 has units "units1" and is converted to a number with units "units2" and returned to N2. In the third case, N1 is assumed to be in SI units and is converted to a number whose units are expressed by U$ and which is returned to N2. In each case, the units specifiers may be any expression of valid units and the operators "*", "/", "^" and the delimeters "(" and ")". "-" is used to separate "units1" from "units2" and to precede the units in case 3 above. If the units

expression cannot be interpreted by the program, it will halt with the error "Signaled Op" (if TRAP(IVL)=2, then the error will be replaced by a warning and the returned value of N2 will be NaN).

Examples:

CALL CONVERT(1,X,"FT") converts 1 foot to X meters. X = 0.3048.

CALL CONVERT(1,X,"-FT") converts 1 meter to X feet. X = 3.28083989501.

CALL CONVERT(1,X,"MI-KM") converts 1 mile to X kilometers. X = 1.609344.

Units Provided

HP-71 Abbrev.	Name	Multiplicative Conversion Constant	Homogeneous SI Units
ACRE	acre	4046.856422	m^2
ATM	atmosphere	101325	Pa
BAR	bar	100000	Pa
BBL	barrel of petroleum	.1589872949	m^3
BCF	billion standard cubic feet of gas	1195300	kg*mol
BTU	British Thermal Unit	1055.056	J
C	degree Celsius	1	K
CAL	calorie	4.1868	J
CM	centimeter	.01	m
CP	centipoise	.001	Pa*s
CST	centistoke	.000001	m^2/s
D	darcy	9.869233E-13	m^2
DAY	day	86400	s
DYNE	dyne	.00001	N
ERG	erg	.0000001	J
F	degree Fahrenheit	.555555555556	K
FT	foot	.3048	m
FTHOH	foot of water	2988.98	Pa
G	gram	.001	kg
GAL	gallon (U.S.)	.003785411784	m^3
GALUK	gallon (U.K.)	.004546087	m^3
HP	horsepower (550 ft*lbf/s)	745.69987	W
HR	hour (mean solar)	3600	s
IN	inch	.0254	m
INHG	inch of mercury (60 F)	3376.85	Pa
INHOH	inch of water (60 F)	248.84	Pa
J	joule	1	J
K	Kelvin	1	K
KCAL	kilocalorie (IST)	4186.8	J
KG	kilogram	1	kg
KGF	kilogram force	9.80665	N
KIP	kilopound force	4448.221615	N
KJ	kilojoule	1000	J
KM	kilometer	1000	m
KMOL	kilomole	1000	mol
KPA	kilopascal	1000	Pa
KSI	kip per square inch	6894757.2	Pa
KT	kilotonne	1000000	kg
KW	kilowatt	1000	W
L	liter	.001	m^3
LBF	pound force	1	N
LBM	pound mass	.45359237	kg
M	meter	1	m
MBAR	millibar	100	Pa
MCF	thousand standard cubic feet of gas	1.1953	kg*mol
MD	millidarcy	9.869233E-16	m^2
MG	megagram	1000	kg

MI	mile	1609.344	m
MIN	minute	60	s
MJ	megajoule	1000000	J
ML	milliliter	.000001	m^3
MM	millimeter	.001	m
MMCF	million standard cubic feet of gas	1195.3	kg*mol
MMHG	millimeters of mercury (60 F)	133.3224	Pa
MN	meganewton	1000000	N
MO	month	2628000	s
MOL	mole	1	mol
MPA	megapascal	1000000	Pa
MT	megatonne	1000000000	kg
MW	megawatt	1000000	W
N	newton	1	N
P	poise	.1	Pa*s
PA	Pascal	1	Pa
PSF	pound force per square foot	47.88025898	Pa
PSI	pound force per square inch	6894.757293	Pa
R	degree Rankine	.555555555556	K
S	second	1	s
SCF	standard cubic foot (60 F, 14.696 psi)	.0011953	kg*mol
SCM	standard cubic meter (15 C, 101.325 kPa)	.0422932	kg*mol
SCMZ	standard cubic meter (0 C, 101.325 kPa)	.0446158	kg*mol
SPGR	specific gravity relative to water (60 F)	999.0412333	kg/m^3
ST	stoke	.0001	m^2/s
T	tonne (metric ton)	1000	kg
THERM	10000 Btu	105505600	J
TON	short ton (2000 lbm)	907.18474	kg
TONUK	long ton (2240 lbm)	1016.046909	kg
TORR	torr (0 C)	133.322	Pa
UM	micrometer	.000001	m
W	watt	1	W
YD	yard	.9144	m
YR	year	31536000	s

This list of units was extracted from the HP-41C Petroleum Fluids Pac.

Listing

```
10 SUB FEETINCH(F1,F2,F$)
20 F=IP(F1)
30 I=ABS(IP(FP(F1)*12))
40 F0=ABS(FP(F1*12)*IP(F2))
50 IF ABS(F2-F0)<=.5 THEN I=I+1 @ F0=0
60 IF I>=12 THEN I=I-12 @ F=F+SGN(F)
70 D$=PEEK$("2F6DC",2) @ STD
80 F$=STR$(F)&" "&STR$(I)&" " @ FIX 0
90 F$=F$&STR$(F0)
100 F$=F$[1,LEN(F$)-1]&"/"&STR$(F2)
110 F$=F$[1,LEN(F$)-1] @ POKE "2F6DC",D$
120 END SUB
130 !
140 SUB DECFEET(F$,F)
150 FOR I=LEN(F$) TO 1 STEP -1
160 IF F$[I,I]#" " THEN 'DONE'
170 NEXT I
180 'DONE': F$=F$[1,I]
190 S=SGN(VAL(F$)) @ IF S=0 THEN S=CLASS(S)
200 'SPC1': S1=POS(F$," ") @ IF S1=1 THEN F$=F$[2] @ GOTO 'SPC1'
210 'SPC2': S2=POS(F$," ",S1+1)
220 IF S2=S1+1 THEN S1=S2 @ GOTO 'SPC2'
230 F=VAL(F$)
240 IF S1 THEN F=F+S*VAL(F$[S1])/12
250 IF S2 THEN F=F+S*VAL(F$[S2])/12
260 END SUB
270 !
280 SUB FEETADD(F1$,F2$,F0,F3$)
290 CALL DECFEET(F1$,F1)
300 CALL DECFEET(F2$,F2)
310 CALL FEETINCH(F1+F2,F0,F3$)
320 END SUB
330 !
340 SUB FEETSUB(F1$,F2$,F0,F3$)
350 CALL DECFEET(F1$,F1)
360 CALL DECFEET(F2$,F2)
370 CALL FEETINCH(F1-F2,F0,F3$)
380 END SUB
390 !
400 SUB FEETMULT(F1$,F2$,F0,F3$)
410 CALL DECFEET(F1$,F1)
420 CALL DECFEET(F2$,F2)
430 CALL FEETINCH(F1*F2,F0,F3$)
440 END SUB
450 !
460 SUB FEETDIV(F1$,F2$,F0,F3$)
470 CALL DECFEET(F1$,F1)
480 CALL DECFEET(F2$,F2)
490 CALL FEETINCH(F1/F2,F0,F3$)
500 END SUB
510 !
520 SUB CONVERT(N1,N2,U$)
530 GOSUB 'INIT'
540 GOSUB 'ZAP'
550 !
560 FOR I=1 TO LEN(U$)+1
```

```
570 IF POS(D$,U$[I,I]) THEN GOSUB 'DELIMS' @ F=0 ELSE F=1
580 IF U$[I,I]="-" THEN GOSUB 'DELIMS' @ K=2 @ V$(1)=V$(1)[1,LEN(V$(1)
)-1]
590 NEXT I
600 !
610 IF F THEN GOSUB 'DELIMS'
620 IF U$="F" OR U$[1,2]="F-" THEN N1=N1+459.67
630 IF U$="C" OR U$[1,2]="C-" THEN N1=N1+273.15
640 IF V$(2)="" THEN V$(2)="1"
650 IF V$(1)="" THEN V$(1)="1"
660 N2=N1*VAL(V$(1))/VAL(V$(2))
670 IF U$[LEN(U$)-1][1,2]="-F" THEN N2=N2-459.67
680 IF U$[LEN(U$)-1][1,2]="-C" THEN N2=N2-273.15
690 GOSUB 'DEINIT'
700 END
710 !
720 'INIT': INTEGER F,I,J,K,O,P
730 O=FLAG(-16,1)
740 DIM C$[486],D$[15],D1$[2],V$(2)[250],X$[5]
750 D1$=PEEK$("2F6DC",2) @ STD
760 D$="0123456789*/^()"
770 ON ERROR GOTO 'ER'
780 X$=ADDR$("CONST")
790 X$=ADDR$("NAMES")
800 GOTO 'OK'
810 'ER': IF ERRN=57 THEN CALL "MAKE" ELSE BEEP 1000 @ DISP ERRM$ @ ST
OP
820 'OK': OFF ERROR
830 ASSIGN #1 TO "CONST"
840 ASSIGN #2 TO "NAMES"
850 READ #2;C$
860 ASSIGN #2 TO *
870 J=1 @ K=1
880 RETURN
890 !
900 'ZAP': FOR I=1 TO LEN(U$)
910 'MORE': IF U$[I,I]=" " THEN U$[I,I]="" @ GOTO 'MORE'
920 NEXT I
930 RETURN
940 !
950 'DELIMS': P=(POS(C$,","&U$[J,I-1]&",")-1) DIV 6
960 READ #1,P;C
970 IF CLASS(P)=-1 THEN C=NAN
980 IF U$[J,I-1]="" THEN C=1
990 V$(K)=V$(K)&STR$(C)&U$[I,I]
1000 'D': I=I+1 @ IF POS(D$,U$[I,I]) THEN V$(K)=V$(K)&U$[I,I] @ GOTO '
D'
1010 J=I
1020 RETURN
1030 !
1040 'DEINIT': POKE "2F6DC",D1$ @ O=FLAG(-16,O) @ RETURN
1050 END SUB
1060 !
1070 SUB MAKE
1080 O=FLAG(-16,1)
1090 ON ERROR GOTO 'E1'
1100 CREATE DATA CONST,81,8
```

```
1110 'E1': ON ERROR GOTO 'E2'
1120 CREATE TEXT NAMES
1130 'E2': OFF ERROR
1140 ASSIGN #1 TO CONST
1150 ASSIGN #2 TO NAMES
1160 DIM C$[486],C(81),B$[5]
1170 C$=",ACRE,,ATM,,,BAR,,,BBL,,,BCF,,,BTU,,,C,,,,,CAL,,,CM,,,,CP,,,"
1180 C$=C$&",CST,,,D,,,,,DAY,,,DYNE,,ERG,,,F,,,,,FT,,,,FTHOH,G,,,,,GAL,,"
1190 C$=C$&",GALUK,HP,,,,HR,,,,IN,,,,INHG,,INHOH,J,,,,,K,,,,,KCAL,,KG,,,"
1200 C$=C$&",KGF,,,KIP,,,KJ,,,,KM,,,,KMOL,,KPA,,,KSI,,,KT,,,,KW,,,,L,,,,"
1210 C$=C$&",LBF,,,LBM,,,M,,,,,MBAR,,MCF,,,MD,,,,MG,,,,MI,,,,MIN,,,MJ,,,"
1220 C$=C$&",ML,,,,MM,,,,MMCF,,MMHG,,MN,,,,MO,,,,MOL,,,MPA,,,MT,,,,MW,,,"
1230 C$=C$&",N,,,,,P,,,,,PA,,,,PSF,,,PSI,,,R,,,,,S,,,,,SCF,,,SCM,,,SCMZ,"
1240 C$=C$&",SPGR,,ST,,,,T,,,,,THERM,TON,,,TONUK,TORR,,UM,,,,W,,,,,YD,,,
,YR,,,"
1250 DATA 4046.856422,101325,1E5,.1589872949,1195300,1055.056,1
1260 DATA 4.1868,.01,.001,1E-6,9.869233E-13,86400,1E-5,1E-7,5/9,.3048,
2988.98
1270 DATA .001,3.785411784E-3,4.546087E-3,745.69987,3600,.0254,3376.85
,248.84
1280 DATA 1,1,4186.8,1,9.80665,4448.221615,1E3,1E3,1E3,1E3,6894757.2,1
E6,1E3
1290 DATA .001,4.44821615,.45359237,1,100,1.1953,9.869233E-16,1E3,1609.
344,60
1300 DATA 1E6,1E-6,.001,1195.3,133.3224,1E6,2628E3,1,1E6,1E9,1E6,1,.1,1
1310 DATA 47.88025898,6894.757293,5/9,1,1.1953E-3,4.22932E-2,4.46158E-2
1320 DATA 999.0412333,1E-4,1E3,105505600,907.18474,1016.046909,133.322
1330 DATA 1E-6,1,.9144,3.1536E7
1340 READ C()
1350 PRINT #2;C$ @ ASSIGN #2 TO *
1360 PRINT #1;C() @ ASSIGN #1 TO *
1370 O=FLAG(-16,O)
1380 END SUB
```

The purpose of these routines is to provide a mechanism by which the volume label of a mini data cassette may be accessed and by which information about the device type and location of the PRINTER IS device may be obtained.

Using the Subprograms

Each subprogram is intended to be used in a fashion similar to that of a standard BASIC keyword. Once a subprogram is loaded into HP-71 memory, it can be called from the keyboard or from a user's program.

EXAMPLE:

From the keyboard,

```
DIM V$
CALL VOLUME(":TAPE",V$)
V$                          displays volume label of tape
```

In a program,

```
10 DIM V$
20 INTEGER I,N
30 STD
40 INPUT "Number of drives ? ";N
50 FOR I=1 TO N
60 CALL VOLUME(":TAPE("&STR$(I)&")",V$)
60 DISP "Drive #";I;': ".'&V$&'"'
70 NEXT I
80 END
```

The subprograms do little or no error checking. Those programs that take parameters assume that their values are correct. Neither is there any error trapping: those errors that occur will suspend the program.

Loading the Subprograms.

Loading the subprograms into the HP-71 is a straight-forward operation which involves copying the code from the provided listing into your computer. It must be stressed that the code as it appears in HP-71 memory must, in most instances, exactly match the provided listing. This is especially true with the POKE statement. The parameters for POKE must be identical to those given in the listing or disastrous results (including a memory reset) may occur.

The listing contains 3 different and independent subprograms. They have been written so that the user may extract any subprogram or group of subprograms desired. All that is involved is to identify the desired subprogram(s) and copy the code from SUB to END SUB. The line numbering is not important, as long as the lines are ordered in the computer as they are in the listing.

Subprogram Descriptions

FINDPR(D$)	This subprogram locates the PRINTER IS device and returns its address to D$. If the PRINTER IS device is on the HP-IL, D$ will be the device's numeric address. Special cases that are also identified are when PRINTER IS "*", "NULL" and "LOOP".
VOLUME(D$,N$)	This subprogram queries drive D$ for the volume label of its cassette. N$ must have a character dimension of at least 6 or the program will be suspended by a "String Ovfl" error. D$ must be a valid mass storage identifier.
VOLUMEIS(D$,N$)	This subprogram writes the volume label N$ to the cassette in drive D$. The characters in N$ should be limited to uppercase alphabet otherwise standard HP-71 statements will not be able to access the new volume label. Only the first 6 characters of N$ will be used. D$ must be a valid mass storage identifier.

"TAPEVOL"

Listing

```
10 DIM D$,P,X$
20 CALL FINDPR(X$)
30 DISP "PRINTER IS :";X$
40 IF X$[1,4]#"LOOP" AND X$#"*" AND X$#"NULL" THEN DISP "AID ";DEVAID(
X$)
50 D$=PEEK$("2F958",2)
60 D$=D$[2]&D$[1,1] @ P=HTD(D$) @ IF P=0 THEN P=INF
70 DISP "PWIDTH ";P
80 END
90 !
100 SUB FINDPR(D$)
110 DIM A$[3],D1$[2] @ REAL A,L @ A=0
120 D1$=PEEK$("2F6DC",2) @ STD
130 A$=PEEK$("2F7AC",1)
140 IF BIT(HTD(A$),3) THEN D$="*" @ GOTO 'END'
150 RESTORE IO
160 PRINT "";
170 A$=PEEK$("2F794",3)
180 IF A$="00F" THEN D$="NULL" @ GOTO 'END'
190 IF A$="FFF" THEN D$="*" @ GOTO 'END'
200 ! This is either an address or LOOP
210 A$=A$[3]&A$[2,2]&A$[1,1] @ A=HTD(A$)
220 L=A DIV 1024+1
230 IF A$[2]="00" THEN 'LOOP'
240 A=BINAND(A,31)+BINAND(A,992) DIV 32/100
250 IF L>1 THEN D$=STR$(A)&":"&STR$(L) ELSE D$=STR$(A) @ GOTO 'END'
260 'LOOP': IF L>1 THEN D$="LOOP:"&STR$(L) ELSE D$="LOOP"
270 'END': POKE "2F6DC",D1$
280 END SUB
290 !
300 SUB VOLUME(D$,N$)
310 A=DEVADDR(D$)
320 IF A=-1 OR DEVAID(D$)#16 THEN N$="" @ END
330 SEND LISTEN A DDL 4 MTA DATA 0,0
340 GOSUB 'STATUS'
350 SEND UNL TALK A DDT 2 UNT UNL
360 GOSUB 'STATUS'
370 ENTER :A USING "#,8A";N$
380 N$=N$[3,8]
390 SEND LISTEN A DDL 7
400 GOSUB 'STATUS'
410 END
420 !
430 'STATUS': S=SPOLL(A)
440 IF BIT(S,5) THEN 'STATUS'
450 IF S THEN BEEP 1000 @ DISP "DRIVE ERROR" @ PAUSE
460 RETURN
470 END SUB
480 !
490 SUB VOLUMEIS(D$,N$)
500 A=DEVADDR(D$)
510 IF A=-1 OR DEVAID(D$)#16 THEN END
520 SEND UNL LISTEN A MTA DDL 4 DATA 0,0
530 GOSUB 'STATUS'
540 SEND UNL TALK A DDT 2
550 GOSUB 'STATUS'
```

```
560 ENTER :A USING "#,8A";R$
570 N$=UPRC$(N$)
580 R$[3,8]=(N$&"      ")[1,6]
590 SEND UNL LISTEN A MTA DDL 1 DATA 0,0
600 GOSUB 'STATUS'
610 SEND UNL LISTEN A MTA DDL 2 DATA R$[1,7] END NUM(R$[8])
620 GOSUB 'STATUS'
630 SEND LISTEN A DDL 7
640 GOSUB 'STATUS'
650 END
660 !
670 'STATUS': S=SPOLL(A)
680 IF BIT(S,5) THEN 'STATUS'
690 IF S THEN BEEP 1000 @ DISP "DRIVE ERROR" @ PAUSE
700 RETURN
710 END SUB
```

The purpose of this program is to provide a listing of all variables contained in a program, and their respective line numbers. Such a listing is very useful in the debugging stages of program development.

Loading the Program.

Loading the program into the HP-71 is a straight-forward operation which involves copying the code from the provided listing into your computer. It must be stressed that the code as it appears in HP-71 memory must, in most instances, exactly match the provided listing. This is especially true with the POKE statement. The parameters for POKE must be identical to those given in the listing or disastrous results (including a memory reset) may occur.

Program Description

The program, once begun (via RUN), prompts the user for the name of the text file to be referenced. This source file must be TEXT and, therefore, the user must TRANSFORM the BASIC program of interest into TEXT before the variable cross reference is performed. This text file need not reside in :MAIN RAM. It may exist on :TAPE or in a :PORT.

The program next prompts for the desired width of the PRINTER IS device. A proper value for this input is the physical width of the output device. This number is necessary for proper formatting of the cross reference output.

Once these two inputs have been given, the program will continue for what may be several minutes before generating output. Subprogram cross references will be separated from main program references. Variable types are not specified except by "$" for string variables, "(" for 1 and 2 dimensional arrays and "$(" for string arrays.

Listing

```
10 DESTROY F$,P
20 INPUT "TEXT FILE ? ";F$
30 INPUT "PRINTER WIDTH ? ";P
40 P$=PEEK$("2F958",2)
50 PWIDTH P
60 DISP "WORKING"
70 FOR I=1 TO P @ PRINT "="; @ NEXT I
80 PRINT "Variable Cross Reference";
90 IF P>25+LEN(F$) THEN PRINT TAB(P-LEN(F$));F$ ELSE PRINT @ PRINT TAB
((P-LEN(F$))/2);F$
100 FOR I=1 TO P @ PRINT "="; @ NEXT I @ PRINT
110 PWIDTH P DIV 5*5
120 CALL VARIABLE(F$)
130 POKE "2F958",P$ @ PUT "#38"
140 END
150 !
160 SUB VARIABLE(F$)
170 ASSIGN #255 TO F$
180 DIM L$[100],L0$[4],A$[26],D$[23]
190 INTEGER I,J,K,L,M,O
200 O=FLAG(-16,1) @ F0=FLAG(0)
210 D1$=PEEK$("2F6DC",2) @ STD
220 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
230 D$="+-*/()<>[]/\?#@%^& :;=,"
240 J=0
250 !
260 'NEXTLINE': ON ERROR GOTO "ERROR" @ READ #255;L$ @ OFF ERROR
270 L$=UPRC$(L$) @ I=6 @ L0$=L$[1,4]
280 !
290 'NEXTCHAR':
300 IF L$[I,I]="!" OR L$[I][1,4]="REM " THEN 'NEXTLINE'
310 IF L$[I][1,4]="SUB " AND L$[I-2,I-2]#"D" THEN GOSUB 'SBPRGM'
320 IF L$[I,I]="'" OR L$[I,I]='"' THEN I=POS(L$,L$[I,I],I+1)+1
330 IF NOT (POS(A$,L$[I,I]) AND NOT POS(A$,L$[I+1,I+1])) THEN 'REPEAT'
340 K=I
350 'VARS': IF NOT POS(D$,L$[I,I]) AND I<=LEN(L$) THEN I=I+1 @ GOTO 'V
ARS'
360 SFLAG 0
370 FOR L=1 TO J
380 IF N$(L)=L$[K,I-(L$[I,I]#"(")] THEN 'OUT'
390 NEXT L
400 CFLAG 0
410 'OUT': IF FLAG(0,0) THEN 'PUTIN'
420 J=J+1
430 DIM N$(J)[4]
440 N$(J)=L$[K,I-(L$[I,I]#"(")]
450 CREATE TEXT "VAR"&STR$(L)
460 ASSIGN #L TO "VAR"&STR$(L)
470 'PUTIN': PRINT #L;CHR$(VAL(L0$[1,2]))&CHR$(VAL(L0$[3]))
480 'REPEAT': IF NOT POS(D$,L$[I,I]) AND I<=LEN(L$) THEN I=I+1 @ GOTO
'REPEAT'
490 I=I+1
500 IF I<=LEN(L$) THEN 'NEXTCHAR' ELSE 'NEXTLINE'
510 'SBPRGM': IF J THEN GOSUB "PRINT"
520 PRINT "-------" @ PRINT STR$(VAL(L0$))&" "&L$[6] @ PRINT
530 RETURN
```

```
540 'PRINT': IF J=0 THEN RETURN
550 FOR M=1 TO J @ ASSIGN #M TO * @ NEXT M
560 CALL PRINTOUT(N$(),J)
570 RETURN
580 'ERROR':
590 IF ERRN#54 THEN DISP ERRM$ @ BEEP 1000,.25 @ OFF ERROR ELSE GOSUB
'PRINT'
600 O=FLAG(-16,O) @ F0=FLAG(0,F0) @ POKE "2F6DC",D1$
610 END SUB
620 !
630 SUB PRINTOUT(N$(),L)
640 DIM A$[2],B$[2],T$[4]
650 INTEGER I,J,K
660 FOR I=1 TO L
670 T$="x"
680 FOR J=1 TO L
690 IF N$(J)<T$ THEN T$=N$(J) @ K=J
700 NEXT J
710 N$(K)="x" @ B$=""
720 IF T$#"x" THEN PRINT T$
730 ASSIGN #K TO "VAR"&STR$(K)
740 'MORE': ON ERROR GOTO "OUT" @ READ #K;A$ @ OFF ERROR
750 IF A$#B$ THEN PRINT USING "#,XDDDD";NUM(A$)*100+NUM(A$[2])
760 B$=A$
770 GOTO "MORE"
780 'OUT': ASSIGN #K TO *
790 PURGE "VAR"&STR$(K)
800 PRINT @ PRINT
810 NEXT I
820 L=0
830 END SUB
```

The purpose of this subprogram is to provide a catalog of all LEX files and their entries. Since the HP-71 system ROM's are treated as LEX files, they are included in the catalog.

Loading the Program.

Loading the subprogram into the HP-71 is a straight-forward operation which involves copying the code from the provided listing into your computer. It must be stressed that the code as it appears in HP-71 memory must, in most instances, exactly match the provided listing.

Subprogram Description

The program will first print the LEX file ID: a two digit hexadecimal number. Next, all entries will be printed. The format is as follows:

Keyword ID: a two digit hexadecimal number.

Keyword name

Keyword type: word, function, statement, special word (special words do not have keyword ID's).

If the keyword is a function, it's parameters and their types are listed.

If the keyword is a statement, its legal uses are listed.

The process is repeated for all keywords in every LEX file. Note that not all LEX files contain keywords.

Sample:

```
ID=01
 01 ACS       Function
     Parm1    Numeric
 02 ADDR$     Function
     Parm1    String
 03 ADJABS    Statement
    Legal:    In program
              After IF
              From KBD
```

```
 10 SUB SYSTCAT
 20 !
 30 DEF FNR$(R$)
 40 R1$=""
 50 FOR J=LEN(R$) TO 1 STEP -1
 60 R1$=R1$&R$[J,J]
 70 NEXT J
 80 FNR$=R1$
 90 END DEF
100 !
110 A=HTD(FNR$(PEEK$("2F571",5)))
120 'NEXT': S$=FNR$(PEEK$(DTH$(A),7))
130 IF S$[4,6]#"BFC" THEN A=A+7+HTD(S$[1,3]) @ GOTO 'NEXT'
140 A=A+7
150 !
160 'LEXID': T$=FNR$(PEEK$(DTH$(A),11))
170 PRINT "ID="&T$[10,11]
180 C=HTD(T$[1,5])
190 M=HTD(T$[8,9])
200 B=C-13
210 B=B+HTD(FNR$(PEEK$(DTH$(B),4)))
220 'LEXENTRY': A$=PEEK$(DTH$(B),16)
230 IF A$[1,2]="FF" THEN 'EOF' ! test for end of LEX file
240 W$="" ! word name
250 L=HTD(PEEK$(DTH$(B-1),1))+1 ! # chrs in name
260 !
270 FOR I=1 TO L STEP 2 ! build name
280 W$=W$&CHR$(HTD(FNR$(A$[I,I+1])))
290 NEXT I
300 !
310 S$=FNR$(PEEK$(DTH$(B+L),2)) ! entry type
320 IF S$="00" THEN PRINT " -- ";W$;TAB(14);"Spec Word/FFN" @ GOTO 'NX
TENTRY'
330 PRINT " ";S$;" ";W$;
340 D=C+(HTD(S$)-M)*9
350 E=HTD(PEEK$(DTH$(D+8),1))
360 W$="Statement"
370 IF E=15 THEN W$="Function" ELSE IF E=0 THEN W$="Word"
380 PRINT TAB(14);W$
390 IF E=15 THEN GOSUB 'FATTRIB' ELSE IF E#0 THEN GOSUB 'SATTRIB'
400 'NXTENTRY': B=B+L+3
410 PRINT
420 GOTO 'LEXENTRY'
430 !
440 'FATTRIB': E=D+1+HTD(FNR$(PEEK$(DTH$(D+3),5)))
450 IF E>1048575 THEN E=E-1048576
460 P=HTD(PEEK$(DTH$(E),1))
470 Q=HTD(PEEK$(DTH$(E+1),1))
480 !
490 FOR I=1 TO Q
500 PRINT "     Parm";STR$(I);TAB(14);
510 E=E-1
520 X=HTD(PEEK$(DTH$(E),1))
530 IF I>P THEN PRINT "Opt ";
540 IF X>11 THEN PRINT "Num or String"; ELSE IF X>7 THEN PRINT "Numeri
c"; ELSE PRINT "String";
```

```
550 IF MOD(X,4) THEN PRINT " Array";
560 PRINT
570 NEXT I
580 RETURN
590 !
600 'SATTRIB': PRINT "    Legal:";
610 IF E>8 THEN PRINT TAB(14);"In program";
620 IF MOD(E,8) DIV 4 THEN PRINT TAB(14);"After IF";
630 IF MOD(E,2) THEN PRINT TAB(14);"From KBD";
640 PRINT
650 RETURN
660 !
670 'EOF': IF T$[10,11]#"00" THEN A=A+11 @ PRINT @ GOTO 'LEXID'
680 END SUB
```

Appendix A.
Memory Locations Accessed via PEEK$ and POKE

address, nibbles - description

2E3FE, 1 - The display contrast setting defined by the CONTRAST statement. The value returned is in the range 0 to F, hexidecimal.

2F471, 4 - The display window as defined by the WINDOW statement. The first byte contains the window start minus 1 (0,21) and the second contains the window length (1,22).

2F571, 5 - Address of main program memory end. Used as a pointer to the file chain.

2F576, 5 - Calc mode stack pointer.

2F580, 15 - Calc mode stack pointers.

2F6D9, 8 - The 32 modifiable system flags (-1 through -32).

2F6E9, 16 - The 64 user flags (0-63).

2F6DB, 1 - The two least significant bits in this nibble contain the option round setting.

2F6DC, 2 - System flags -13 through 20: display format. Note that the lower-case-lock and option-base flags are also located in these nibbles. The bits of nibble 2F6DC represent flags -13 to -16, the least significant mapping onto flag -13 and the most significant mapping onto -16. In the same fashion, the bits of nibble 2F6DD map onto flags -17 to -20. HTD(PEEK$('2F6DD',1)) returns a decimal number from 0 to 11 representing (except in STD mode, where they are not significant) the number of fractional digits to display.

2F794, 3 - PRINTER IS information.

2F7AC, 1 - HP-IL loop status. Bit 3 indicates whether OFF IO has been executed.

2F7AD, 3 - The current statistical array name.

2F946, 4 - The first byte is the character delay in 1/32's of a second. The second byte is the line delay in

1/32's of a second. Values of 255 for either of these represents Inf.

2F94F, 2 - The display width (0-255). If this byte is 0, WIDTH is Inf.

2F958, 2 - The printer width (0-255). If this byte is 0, PWIDTH is Inf.

2F95A, 1 - The number of endline nibbles at the end of PRINT'ed output (2-6).

2F95B, 2-6 - The endline sequence at the end of PRINT'ed output (1-3 characters).

2F976, 1 - The number of command stack entries (1 to 16 represented by 0 to F).

00071-90066

Printed in U.S.A.